CS 121 - Intro to Java - Lecture 18

Announcements

Exam end of next week;
extra credit assignment posted on website;

Next class: CS 187 (CS minor: 121+187 + 3 others;
also, a BA in CS - 11 CS classes + minor or major in
something else..)

There IS class tomorrow, here - Thursday schedule

No office hours today 4-6
public class SizedPerson extends Person{

    private int height; // attribute only in SizedPerson

    public SizedPerson(String who, int ht){
        super(who); // build person part of SizedPerson
        height = ht;
    }

    public void personInfo(){
        super.personInfo(); // personInfo frm super class
        System.out.println("my height is " + height);
    }
}
public class PersonDriver{
    public static void main(String[] args){
        SizedPerson p = new SizedPerson("hilda", 62);
        p.personInfo();
        System.out.println("big name? " + p.bigName());
    }
}

-----------------
Output
-----------------

I am hilda
my height is 62
big name? false
Object

Random

Chance

Drunk
What’s Object?

Object is the ultimate base class.

Every class is either directly or indirectly derived from Object

Object has a few methods - One is `toString`

Example:

```java
Infant myKid = new Infant("jill", 23);
System.out.println(myKid.toString());
```

Prints: `Infant@124821`

Reason: we haven’t overwritten `toString`
Files

How to read from / write to them
Barring catastrophe your computer’s file system is permanent.

It’s the repository of data (your programs, term papers, diary, emails, popular URLs, and so forth)

Actually, we’ll distinguish between text files (blah.txt, Infant.java), for which the basic unit is the character, and binary files, for which the basic unit is the byte.

One other distinction: files can be read in sequential style - beginning to end (very inefficient for some applications); or in random access file (jump directly to a disk location - much more efficient).

Our focus here: sequential text files
import java.util.Scanner;
import java.io.*;

public class DisplayFile{
    public static void main(String[] args) throws IOException{
        Scanner nameReader = new Scanner(System.in);
        System.out.println("Enter a file name");
        String fileName = nameReader.nextLine();
        Scanner scanFile = new Scanner(new File(fileName));

        while(scanFile.hasNextLine()){
            System.out.println(scanFile.nextLine());
        }
        scanFile.close();
    }
}
Scanner scanFile = new Scanner(new File(fileName));

Or

Scanner scanFile = new Scanner(new FileReader(fileName));
Some Observations:

Often what we do when we read a text file is done at the line level (e.g. here’s a line: capitalize it, count it, convert it to numbers, look for a word, etc.)

The file-reading machinery is unchanging: get the name, make a File object (or a FileReader object), make a Scanner, give me the lines

We’ll separate these two aspects of DisplayFile, then use inheritance so that we never have to re-examine the unchanging part!
Second observation:

Java exception mechanism - it’s around; we’ll talk about in the next class
import java.util.Scanner;
import java.io.*;

public class Echo{
  String fileName; // external file name
  Scanner scan; // reads from external file

  public Echo(String f) throws IOException{
    fileName = f;
    scan = 
       new Scanner(new File(fileName));
  }
}
public void readLines(){
    while(scan.hasNext()){
        processLine(scan.nextLine());
    }
    scan.close();
}

public void processLine(String line){
    System.out.println(line);
}

```java
public Echo(String f) throws IOException{
    fileName = f;
    scan =
        new Scanner(new FileReader(fileName));
}

public void readLines(){
    while(scan.hasNext()){
        processLine(scan.nextLine());
    }
    scan.close();
}

Core-Unchanging

void processLine(){..}
to be overwritten in subclass
```
// Echo revisited...

import java.util.Scanner;
import java.io.*;
public class ReadDriver{
    public static void main(String[] args) throws IOException{
        String fileName;
        Scanner nameReader = new Scanner(System.in);
        System.out.println("Enter a file name");
        fileName = nameReader.nextLine();
        Echo e = new Echo(fileName);
        e.readLines();
    }
}
import java.io.*;
public class LineCount extends Echo{
    private int count = 0;

    public LineCount(String f) throws IOException {
        super(f);   // do File drill that Echo does
    }

    public void processLine(String line){count++;}

    public void reportLineCount(){
        System.out.println("Line count: "+ count);
    }
}
LineCount extends Echo

int count = 0;

Echo

String filename

Echo(String f) // file handling here

void readLines() // core code

void processLine(String line){..}

void processLine(String line){count++;}

void reportLineCount(){...}
Write an application that reads an external text file, and echoes the file to the console with line numbers
NumberEcho extends Echo

int lineNum = 0;

Echo
String filename
Echo(String f)
void readLines()
void processLine(String line){..}

void processLine(String line){lineNum++;
    System.out.println(lineNum + " " + line);}
Write an application that reads an external text file, counts the number of characters in the file, and reports this sum as a message to the console.
CharCounter extends Echo

int charCt = 0;

Echo
String filename
Echo(String f)
void readLines()
void processLine(String line){..}

void processLine(String line){charCt += line.length();}
void reportChars(){ System.out.println(charCt);}
Write an application that reads an external text file, and then reports to the console the number of words in the file
WordCount

```java
int wordCt = 0;

Echo
String filename
Echo(String f)
void readLines()
void processLine(String line){..}

void processLine(String line){chop up, count, update}
void reportCount(){System.out.println(wordCt);}
```
The Marriage
The problem:

Report the letter frequencies in a text file

The marriage:

The Letters class

(with its scoreboard, etc - Prog #5)

File processing via Echo and its subclasses
The solution:

Subclass Echo to a new class, LetterPat

endow with a Letters object

that object - of course - comes with the functionality of every Letters object.

Note: we DO NOT have to mess with the scoreboard - that’s under the hood
The driver...

```java
import java.util.Scanner; import java.io.*;
public class LetterDriver{
    public static void main(String[] args) throws IOException
    {
        String fileName;
        Scanner nameReader = new Scanner(System.in);
        System.out.println("Enter a file name");
        fileName = nameReader.nextLine();
        LetterPat e = new LetterPat(fileName);
        e.readLine(); // calls processLine, as usual
        e.reportLetters();
    }
}
```
The API for my Letters class..

Letters - constructor of no arguments

process(String s) - processes a String, services the scoreboard

reportLetters() - prints the chart of letter freqs

I DO NOT need to know about the scoreboard - the Letters class handles that! (the principle is: “I don’t know and I don’t care”)

import java.io.*;

public class LetterPat extends Echo{
    private Letters letters = new Letters();

    public LetterPat(String f) throws IOException {
        super(f); // act like an Echo object
    }

    public void processLine(String s) {
        // s: line of text
        letters.process(s); // walks down s, updating scoreboard
    }

    public void reportLetters() {
        letters.reportLetters();
    }
}

import java.io.*;
import java.util.*;
public class WriteFile{
    public static void main(String[] args) throws IOException{
        System.out.println("Enter fname-will hold output");
        Scanner nameReader = new Scanner(System.in);
        String fileName = nameReader.nextLine();
        PrintWriter writer = new PrintWriter(fileName);
        Scanner scan = new Scanner(System.in);
        String s = " "; // a String of length 1
        System.out.println("Enter text, end with 2 returns");
        while(s.length() > 0){
            s = scan.nextLine();
            writer.println(s);
        }
    }
}
// WriteFile.. Continued

writer.close();

// now echo the file back to the console
Echo e = new Echo(fileName);
System.out.println("Here comes the echo");
System.out.println();
e.readLines();
}
import java.util.Scanner; import java.io.*;

public class Transform extends Echo{
    String inName, outName; // external file names
    Scanner scan; // Scanner object for reading external file
    PrintWriter writer;

    public Transform(String inF,String outF)throwsIOException
    {
        super(inF);
        inName = inF;
        outName = outF;
        scan = new Scanner(new FileReader(inName));
        writer = new PrintWriter(outF);
    }
}
public void readLines(){ // reads lines, hands each to processLine
    super.readLines();
    writer.close();
}

public void processLine(String line){
    /* hands work off to embedded PrinterWriter object writer */
    writer.println(line);
}
}
Our next example illustrates a bizarre anecdote that circulated on the web several years ago, and caused widespread giggles among all who saw it. The story concerns an experiment by British researchers that put forward the following argument: it turns out that if you take a single body of text, extract each word, leave the first two and last two letters of every word unchanged, but then randomly rearrange the rest - that is, the middle, then the ensuing text is completely unrecognizable.
Ok, here’s the deal:

Each word is distorted as follows:

- first two, last two characters left undisturbed
- middle characters - if there are any - are randomly permuted
- then result is pasted into a line and printed out
void processLine(String line){
    chop line into words;
    if word is short, paste on answer, else:  break apart - scramble middle - reassemble, paste on answer}

void readLines()

Echo

String filename

Echo(String f)
import java.util.*;
import java.io.*;
public class ScrambleDriver{
    public static void main(String[] args) throws IOException {
        Scanner scan = new Scanner(System.in);
        System.out.println("enter file name");
        String fileName = scan.next();
        TextScramble scram = new TextScramble(fileName);
        scram.readLines();
    }
}
import java.io.*;
import java.util.*;
public class TextScramble extends Echo{
    Random r = new Random();
    String blank = " ";

    public TextScramble(String f) throws IOException{
        super(f);
    }
}
public void processLine(String line){
    String s = blank;
    String answer = ""; // will hold full line-reassembled
    char[][] mid; // holds middle of string, for scrambling
    StringTokenizer str = new StringTokenizer(line);
        while(str.hasMoreTokens()){
            s = str.nextToken();
            if (s.length() < 6) answer = answer + s + blank;
            else
                { do the real work }
        }
    System.out.println(answer);
}
public void processLine(String line){
    String s = blank; String answer = ""; char[] mid;
    StringTokenizer str = new StringTokenizer(line);
    while (str.hasMoreTokens()){
        s = str.nextToken();
        if (s.length() < 6) answer = answer + s + blank;
        else { // carve up the token..
            String front = s.substring(0,2);
            String back = s.substring(s.length() - 2, s.length());
            String middle = s.substring(2,s.length() -2);
            mid = middle.toCharArray(); // make middle an array
            shuffle(mid);
            middle = new String(mid); // transform back to String
            answer = answer + front + middle + back + blank;
        }
    }
    System.out.println(answer);
}
if (s.length() < 6) answer = answer + s + blank;

else {
    String front = s.substring(0, 2);
    String back = s.substring(s.length() - 2, s.length());
    String middle = s.substring(2, s.length() - 2);
    mid = middle.toCharArray();
    shuffle(mid);
    middle = new String(mid);
    answer = answer + front + middle + back + blank;
}