CS1007 lecture #20 notes

tue 27 nov 2001

http://www.cs.columbia.edu/~sklar/cs1007

today:
- news
- input and output, aka I/O (ch 8, especially 8.2-8.4)
- StringTokenizer

news.

- my office hours
  - this week only: Thu office hours from 4.15pm-5.00pm (instead of 12.30pm)
  - sign up for office hours outside my office door
- final exam
  - official exam: Tue Dec 18 9am-12noon
  - makeup exam: Thu Dec 20 1pm-4pm
  - you MUST sign up for the makeup exam — sign up sheet is being passed around in class today
- homework #7 will be posted today, due Wed Dec 5
  there will be NO extensions!

input and output.

I've drawn a picture of input and output many times this semester.

input → CPU → output

Up to now, input has been from the keyboard and output has been to the screen.

Today we will read input from "text files" and write output to "text files".

files.

We've talked about two kinds of files:

- text files, like *.java files, *.html files, etc
- binary files, like *.class files

You can test the file type by entering:

```
unix$ more <filename>
```

and if your file is not a text file, you'll get back an error like this:

```
******** <filename>: Not a text file ********
```


java.io package.

Typically, there are three processing steps when using files:

1. open
2. read, write or update
3. close

We'll only talk about read and write in Java.

In order to implement file I/O in Java, we need several classes from the java.io package:

- FileReader
- FileWriter
- BufferedReader
- BufferedWriter
- PrintWriter

programs that work with data files.

The simple model for programs that work with data files is to:

1. open the data file for reading
2. read the contents of the data file into program variables
3. close the data file
4. manipulate the values in the program variables
5. open the data file for writing
6. write the manipulated values to the data file
7. close the data file

We'll use the Inventory example in the textbook (listing 8.7) and expand on that.

InvItem class (based on listing 8.8 from the textbook).

```
public class InvItem {

  private String name;
  private int    units;
  private float  price;

  public InvItem( String itemName, int numUnits, float cost ) {
    name  = itemName;
    units = numUnits;
    price = cost;
  } // end of InvItem constructor

  public String getName() {
    return( name );
  } // end of getName() method

  public int getUnits() {
    return( units );
  } // end of getUnits() method

  public float getPrice() {
    return( price );
  } // end of getPrice() method

  public void setPrice( float new_price ) {
    price = new_price;
  } // end of setPrice() method

} // end of InvItem class
```


Inventory class (based on listings 8.7 and 8.9 from the textbook).

```
import java.util.*;
import java.io.*;

public class Inventory {

  public static void main( String[] args ) {

    final int MAX = 100;
    InvItem[] items = new InvItem[MAX];
    StringTokenizer tokenizer;
    String line, name, filename="inventory.dat";
    int units, count=0;
    float price;
```



```
    // read data from file into program variables
    try {
      FileReader fr = new FileReader( filename );
      BufferedReader infile = new BufferedReader( fr );
      line = infile.readLine();
      while( line != null ) {
        tokenizer = new StringTokenizer( line );
        name = tokenizer.nextToken();
        try {
          units = Integer.parseInt( tokenizer.nextToken() );
          price = Float.parseFloat( tokenizer.nextToken() );
          items[count++] = new InvItem( name,units,price );
        }
        catch( NumberFormatException nfx ) {
          System.out.println( "error in input; line ignored: " + line );
        }
        line = infile.readLine();
      } // end of while
      infile.close();
    }
    catch( FileNotFoundException fnfx ) {
      System.out.println( "file not found: " + filename );
    }
    catch( IOException iox ) {
      System.out.println( iox );
    }
```

```
    // manipulate the values of the program variables:
    // increase the prices by 10%
    for ( int i=0; i<count; i++ ) {
      items[i].setPrice( items[i].getPrice()*1.10 );
    } // end for i


    // display the values of the program variables on the screen
    for ( int i=0; i<count; i++ ) {
      System.out.println( items[i].getName() +
                          " number of items = " + items[i].getUnits() +
                          " price = $" + items[i].getPrice() );
    } // end for i
```

```
    // write data from program variables into data file
    try {
      FileWriter fw = new FileWriter( filename );
      PrintWriter outfile = new PrintWriter( new BufferedWriter( fw ));
      for ( int i=0; i<count; i++ ) {
        outfile.println( items[i].getName() + " " +
                         items[i].getUnits() + " " +
                         items[i].getPrice() );
      } // end for i
      outfile.close();
    }
    catch( FileNotFoundException fnfx ) {
      System.out.println( "file not found: " + filename );
    }
    catch( IOException iox ) {
      System.out.println( iox );
    }

  } // end of main() method

} // end of Inventory class
```

StringTokenizer

- used to break up a string into “tokens”, i.e. components
- each token is separated by a “delimiter”
- default delimiter is whitespace
- but you can set another value for delimiter
- primary method used: `public String nextToken();`

example run:

data file (inventory.dat) =

```
Widget 14 3.35
Spoke 132 0.32
Wrap 58 1.92
Thing 28 4.17
```

first time through the while loop in the main() method of Inventory:

line = `"Widget 14 3.35"`

name = `"Widget"`

unit → `Integer.parseInt( "14" ) = 14`

price → `Float.parseFloat( "3.35" ) = 3.35`