CS 121 - Intro to Programming: Java - Lecture 18

Announcements

OWL for ch 8 due tonight
ProSix due 4/5, Ch 9 OWL due 4/7

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

Inheritance revisited

Remember: Inheritance is a mechanism for dealing with the brittle nature of object model.

Supports recycling, encapsulation
public class Person{
    private String name;

    public Person(String who){ name = who;}

    public String getName(){return name;}

    public void personInfo(){
        System.out.println("I am " + name);
    }

    public boolean bigName(){
        if (name.length() > 6)
            return true;
        else
            return false;
    }
}
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
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

just in the base class
personInfo from derived class
Make a Tool class (name, weight);

Extend to a PowerTool class (amps - double)
public class Tool{
    String name;
    double weight;
    
    public Tool(String n, double w){
        name = n;
        weight = w;
    }
}
public class PowerTool extends Tool{
    double amps;
    public PowerTool(String n, double w, double a){
        super(n,w);
        amps = a;
    }
}
Make a Road class (start, finish, distance), include a toString method;

Extend to a MinorRoad class (is it paved, or not) also: add a toString method
public class Road{
    private String start;
    private String finish;
    private int distance;

    public Road(String s, String f, int d){
        start = s; finish = f; distance = d;
    }

    public String toString(){
        String s = "from "+ start +" to " + finish +
        ": " + distance);
    }
}
public class MinorRoad extends Road{
private boolean paved;

    public MinorRoad(String s, String f, int d,
                        boolean p){
        super(s,f,d);
        paved = p;
    }

    String toString(){
        String s = super.toString();
        return(s + " paved? " + paved);
    }
}
Person is the **base** class or the **super** class; SizedPerson is the **derived** or the **subclass** class. SizedPerson **extends** Person; SizedPerson **specializes** Person.

The **super** class is the **more general** class..
int height

String name

Person(…)

getName()

bigName()

personInfo()

Person

SizedPerson(…)

personInfo()

SizedPerson
Suppose I create a class like this:

```java
public class Chance extends Random{...}
```

What do I get??
import java.util.Random;

public class Chance extends Random {
    public int throwDie() { return (1 + nextInt(6)); }

    public int throwDice() {
        return (throwDie() + throwDie());
    }

    public void shuffle(int[] nums) {
        // random perm
        int swapPos, temp;
        for (int i = nums.length - 1; i > 0; i--) {
            swapPos = nextInt(i + 1); // pick pos from 0 -> i
            temp = nums[swapPos]; // swap vals at i, swapPos
            nums[swapPos] = nums[i];
            nums[i] = temp;
        }
    }
}
The Drunk at the Opera

Drunk takes over opera cloak room
Hands out umbrellas at random
Sometimes, nobody gets own umbrella;
Other times some do get their own own.

In 10,000 drunken episodes, how frequently does no one gets his or her own?????
(answer: 1/e (CS575/Math513) or ~.37)
public class Drunk extends Chance{
    private int count;
    private int[] patrons;  // the opera goers

    public Drunk(int count){
        this.count = count;
        patrons = new int[count];
    }

    public void initialize(){  // start: each gets own
        for(int pat = 0; pat < patrons.length; pat++)
            patrons[pat] = pat;
    }
}
public boolean allGetWrong()
{
    boolean allWrong = true;
    initialize(); // everyone gets own
    shuffle(patrons); // drunk works magic
    for(int pat = 0; pat < patrons.length; pat++)
    {
        if( patrons[pat] == pat )
        {
            allWrong = false;
            break;
        }
    }
    return allWrong;
}

import java.util.Scanner;

public class DrunkTester{
    public static void main(String[] args){
        int trials = 10000;
        int allWrong = 0;
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter number of patrons");
        int num = scan.nextInt();
        Drunk d = new Drunk(num);
        for(int j = 0; j < trials; j++){
            if (d.allGetWrong()) allWrong++;
        }
        System.out.println("fraction all wrong umbrellas: " +
                           (double)allWrong/trials);
    }
}

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`

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

Prints: Infant@124821
```

Reason: we haven’t overwritten `toString`

Also: `equals`
Inheritance - a more general view

• Standard subclassing (Apt -> RentedApt)
• Repurposing (Random -> Chance)
• Boosting (hiding stylized details) - we'll do File i.o., graphics display this way
Reminder:

The ArrayList class

(a trapdoor in the text)

ArrayList is an array-like structure with no fixed size
import java.util.*;
public class Backwards2{
    public static void main(String[] args){
        ArrayList<String> lines = new ArrayList<String>();
        Scanner scan = new Scanner(System.in);
        int pos = 0;  String t = " ";  String phrase;
        while(t.length() > 0){
            t = scan.nextLine();  lines.add(t);
        }
        for(int j = lines.size()-1; j >= 0; j--){
            phrase = (lines.get(j)).toUpperCase();
            System.out.println(phrase);
        }
    }
}
## Important operations on ArrayList objects

<table>
<thead>
<tr>
<th>Method</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(e)</td>
<td>appends new element e as last cell</td>
</tr>
<tr>
<td>remove(i)</td>
<td>removes element at location i</td>
</tr>
<tr>
<td>set(i,e)</td>
<td>sets element at position i to e</td>
</tr>
<tr>
<td>get(i)</td>
<td>returns element at position i in the ArrayList</td>
</tr>
<tr>
<td>size()</td>
<td>returns number of elements in ArrayList</td>
</tr>
</tbody>
</table>
A “boosting” example:

We’ll read in lines of text from the keyboard, then print the words in those lines back to the console.

Generally:

We’ll read in lines of text from the keyboard, then do “X” with the words, and report back...

X:

Count the words

Report a unique list of the words

Report the distribution of word sizes

Report the words and their multiplicites
In WordStudy we’ll read in lines, then print words in the entered lines back to the console

then we’ll extend the WordStudy class so that its core - the word extraction from multiple lines - is left intact.
import java.util.*;
public class Study{
    ArrayList<String> list = new ArrayList<String>();

    public void studyWords(String t){
        String s;
        StringTokenizer str = new StringTokenizer(t,",.?!;: ");
        while(str.hasMoreTokens()){  
            s = str.nextToken();
            //process(s); // accounts for each word in line of text
            process(s);
        }
    }

    public void process(String s){list.add(s);}

    public void studyReport(){
        for(String s : list)
            System.out.println(s);
    }
}
studyWords is handed a String, produces tokens via process method - which is meant to be overwritten by extending classes
list - common core data structure

studyWords - common core action

process - handed a token, services list

studyReport - provides some kind of report about the list data structure
import java.util.*;

public class WordStudyDriver{
    public static void main(String[] args){
        Study study = new Study();
        Scanner scan = new Scanner(System.in);
        String t = " ";
        System.out.println("Enter lines of text, type two returns in a row to end");

        while(t.length() > 0){
            t = scan.nextLine(); // read next line
            t = t.toLowerCase(); // convert line to all lower case
            study.studyWords(t); // hands line to studyWords
        }
        study.studyReport();
    }
}

public class CountStudy extends Study{

    public void studyReport(){
        System.out.println("word count: "+
                            list.size());
    }
}

No change to core method studyWords, or process
public class UniqueStudy extends Study{

    public void process(String t){
        if (!belongs(t)) list.add(t);
    }

    public boolean belongs(String s){
        boolean occurs = false;
        for(String r : list)
            if (r.equals(s)) {occurs = true; break;}
        return occurs;
    }
}

process altered; studyWords, studyReport unchanged
Now we want a report of the sizes of words in the supplied text:

1 3

2 5

3 7

4 6  -> means: "6 words of size 4"

...

15 0

16 3
public class SizesStudy extends Study {
    int[] wordSizes = new array[17];

    public SizesStudy() {
        super();
        initWordSizes();
    }

    public void initWordSizes() {
        for (int j = 0; j < wordSizes.length; j++) {
            wordSizes[j] = 0;
        }
    }
}
public void process(String t){
    int wordLen = t.length();
    if (wordLen > 15) wordSizes[16]++;
    else wordSizes[wordLen]++;
}

public void studyReport(){
    for(int j = 1; j <= 16; j++)
        System.out.println(j + " " + wordSizes[j]);
}