More Class Design

Lecture 12, Mon Feb 1 2010

borrowing from slides by Paul Carter and Steve Wolfman

http://www.cs.ubc.ca/~tmm/courses/111-10
Department of Computer Science
Undergraduate Events

Events this week

Resume Editing Drop-In Session
Date: Mon., Feb 1
Time: 11 am – 2 pm
Location: Rm 255, ICICS/CS

EADS Info Session
Date: Mon., Feb 1
Time: 3:30 – 5:30 pm
Location: CEME 1202

Job Interview Practice Session (for non-coop students)
Date: Tues., Feb 2
Time: 11 am – 1 pm
Location: Rm 206, ICICS/CS

RIM Info Session
Date: Thurs., Feb 4
Time: 5:30 – 7 pm
Location: DMP 110

Events next week

Finding a Summer Job or Internship Info Session
Date: Wed., Feb 10
Time: 12 pm
Location: X836

Masters of Digital Media Program Info Session
Date: Thurs., Feb 11
Time: 12:30 – 1:30 pm
Location: DMP 201
Reminders

- Assignment 1 due Wed 5pm
- TA office hours in DLC
- my office hours 4-5pm today in X661
News: Lab Schedule Change

- no labs next week Feb 8-12
- TAs will hold office hours in labs during Monday lab times to answer pre-midterm questions
  - Mon Feb 8 11am - 3pm ICICS 008
- labs resume after break
  - staggered to ensure that even Monday morning labs have seen material in previous week's lecture
Midterm Coverage

- reading: chapters 1-4
- lectures: weeks 0-4
  - through this Friday 2/5
- topics:
  - intro, hardware background, programming languages, comments, identifiers, whitespace, errors, variables, primitive data types, assignment, casting, constants, objects, classes, strings, input, class design
- assignments:
  - assignment 1
Midterm Format

- closed book, no notes, no calculators
- must bring ID, put in front of you face up so we can see picture and name
- 6:30 Monday 2/8, FSC 1005
  - exam *starts* at 6:30, please arrive before that
  - you will have 60 minutes to do the exam
  - do not turn over or open exam until we say to begin
Midterm Advice

- good to read book, but definitely don't stop there!
- best thing to do: practice programming
  - try exercises in Big Java
    - solutions for some practice problems now posted in new Handy Links folder on WebCT Vista
  - and/or invent your own problems!
- do a mix of programming on the computer, and on paper
  - you will only have paper for the exam
Reading Assignment This Week

- Chap 4.3-4.5 re-read
### Recap: UML Visual Syntax

- + for public, - for private
- fields above, methods below

<table>
<thead>
<tr>
<th>Classname</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ field: type</td>
</tr>
<tr>
<td>- field: type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Classname()</td>
</tr>
<tr>
<td>+ method(): return type</td>
</tr>
<tr>
<td>+ method(param1 type, param2 type): return type</td>
</tr>
<tr>
<td>- method(): return type</td>
</tr>
</tbody>
</table>
Recap: Control Flow Between Modules

- Two weeks ago it was easy to understand control flow: order in which statements are executed
  - march down line by line through file
- Now consider control flow between modules

```
Client code
int rollResult;
myDie.setSides();
rollResult = myDie.roll();

Die class methods
public int roll()
{
    ...
}

public void setSides()
{
    ...
}
Recap: UML Design for Point

- preliminary design for 2D point class

<table>
<thead>
<tr>
<th>Classname</th>
</tr>
</thead>
<tbody>
<tr>
<td>- x: double</td>
</tr>
<tr>
<td>- y: double</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Classname(inX: double, inY: double)</td>
</tr>
<tr>
<td>+ distanceBetween(Point otherPoint): double</td>
</tr>
</tbody>
</table>
public class Point {

}
Formal vs. Actual Parameters

- **formal** parameter: in declaration of class
- **actual** parameter: passed in when method is called
  - variable names may or may not match
- if parameter is primitive type
  - **call by value**: value of actual parameter copied into formal parameter when method is called
  - changes made to formal parameter inside method body will not be reflected in actual parameter value outside of method
- if parameter is object: covered later
Scope

- Fields of class are have **class scope**: accessible to any class member
  - in `Die` and `Point` class implementation, fields accessed by all class methods
- Parameters of method and any variables declared within body of method have **local scope**: accessible only to that method
  - not to any other part of your code
- In general, scope of a variable is block of code within which it is declared
  - **block** of code is defined by braces `{ }`
Commenting Code

- Conventions
  - explain what classes and methods do
  - plus anywhere that you've done something nonobvious
    - often better to say why than what
      - not useful
        ```
        int wishes = 3; // set wishes to 3
        ```
      - useful
        ```
        int wishes = 3; // follow fairy tale convention
        ```
javadoc Comments

- Specific format for method and class header comments
  - running javadoc program will automatically generate HTML documentation

- Rules
  - /** to start, first sentence used for method summary
  - @param tag for parameter name and explanation
  - @return tag for return value explanation
  - other tags: @author, @version
  - */ to end

- Running
  - % javadoc Die.java
  - % javadoc *.java
/**
   Sets the die shape, thus the range of values it can roll.
   @param numSides the number of sides of the die
*/
public void setSides(int numSides) {
    sides = numSides;
}

/**
   Gets the number of sides of the die.
   @return the number of sides of the die
*/
public int getSides() {
    return sides;
}
javadoc Class Comment Example

/** Die: simulate rolling a die
 * @author: CPSC 111, Section 206, Spring 05-06
 * @version: Jan 31, 2006
 *
 * This is the final Die code. We started on Jan 24,
 * tested and improved in on Jan 26, and did a final
 * cleanup pass on Jan 31.
 */
Cleanup Pass

- Would we hand in our code as it stands?
  - good use of whitespace?
  - well commented?
    - every class, method, parameter, return value
  - clear, descriptive variable naming conventions?
  - constants vs. variables or magic numbers?
  - fields initialized?
  - good structure?
  - follows specification?

- ideal: do as you go
  - commenting first is a great idea!

- acceptable: clean up before declaring victory
Key Topic Summary

- Generalizing from something concrete
  - fancy name: abstraction
- Hiding the ugly guts from the outside
  - fancy name: encapsulation
- Not letting one part ruin the other part
  - fancy name: modularity
- Breaking down a problem
  - fancy name: functional decomposition