Our CS1 course principles

• Procedural
  – use objects early, define classes late
  – early focus on procedural decomposition into methods

• "Back to Basics"
  – minimize use of scaffolding, pre-written complex code
  – de-emphasize fancy graphics and output
  – focus on algorithmic thinking and problem solving

• challenging, highly structured course
  – assignments and exams: difficult but well-specified, predictable
  – strict grading scale
  – lots of help available (lab hours, section, book, email, msg.board)
Classic procedural tasks are tough!

- **Selection with if/else**
  - nesting, most closely nested rule
  - Are these 3 ints consecutive? What triangle do these points represent?
  - boolean logic: De Morgan's Law, flags, passing/returning boolean

- **Loops**
  - definite vs. indefinite loops
  - loop invariants and logical assertions
  - fencepost and sentinel problems

- **Parameters/return**
  - students do **not** understand either until after lots of practice

- **Arrays**
  - shifting elements
  - using arrays for counting and tallying
  - unfilled arrays as lists; inserting and removing
  - arrays as parameters (references)
Why Java?

• Why not C/C++ or Pascal?
  – C[++] makes too many simple things hard (strings, memory, .h files, pointers, no objects, bad type system, limited libraries)
  – Pascal would be fine for some tasks, but worse for OOP and is no longer an active "real world" language being used
  – Java gives us some nice things that the others don't (collections, I/O abstractions, good community and software support, etc.)

• Why not Python, Ruby, etc.?
  – industry demand for Java
  – proliferation of resources, materials, libraries
  – easier transition to upper-division courses
Course structure

• lectures
  – "sage on the stage" (with lots of questions)

• TA discussion sections
  – 1 per week; 50 minutes
  – led by undergrad TAs
  – each comes with a problem set, syntax cheat sheet, solution key

• closed labs
  – new feature for Spring 2010
  – optional weekly 80-minute lab for 1 credit ("CSE 190")
• **Building Java Programs**, 2nd edition  ([buildingjavaprograms.com](http://buildingjavaprograms.com))
  – new chapters: implementing collections linked lists, binary trees
  – improved: proc. design, boolean logic, arrays, OOP, inheritance
  – new end-of-chapter exercises geared at our labs / TA sections

• textbook as central hub of course
  – weekly written problems (due in section)
  – tight integration between lectures and book
  – case studies similar to weekly assignments
  – end-of-chapter problems solved in lab/section
  – book as reference on our open-book exams
  – same book for CS1 and CS2
  – resources available on book web site
Grading and assessment

• homework 45%, midterm 20%, final 30%, participation 5%

• homework assignments as focal point of course
  – approximately one assignment per week; 60-100 lines of code
  – graded 50/50 on "external" and "internal" correctness (style)
  – detailed problem descriptions
  – vague solution descriptions (not, "implement this interface")
  – provide testing materials (expected outputs; unit tests)
Weeks 1-2

On the 1st day of "Xmas", my true love gave to me a partridge in a pear tree.

On the 2nd day of "Xmas", my true love gave to me two turtle doves, and a partridge in a pear tree.

On the 3rd day of "Xmas", my true love gave to me three French hens, two turtle doves, and a partridge in a pear tree.
Weeks 3-4

• Week 3: Parameterization (Ch. 3; Supplement 3G)
  - 2D graphics (as killer example for parameterization)
  - instructor-provided DrawingPanel class
  - lecture programs: draw cars, pyramids, projectiles, etc.
  - homework: draw a patterned graphical figure or optical illusion

• Week 4: Data/control flow; selection (Ch. 4)
  - return if/else
  - interactive programs with Scanner
  - lecture program: compute TA salaries; body mass index
  - homework: SAT/ACT scores/admissions; Grades; birthdays
Student creativity

- Creative song verses
- ASCII art contest
- Doodle contest
- Haiku poems
- Mad lib creation
- Personality test
- Critter Husky competition
Facebook application

- **FB app** created by former TA/instructor Victoria Kirst
- students post creative homework submissions
- other students and FB friends can vote on favorites
- an excellent advertisement for CSE 142!
Weeks 5-6

<< your haiku intro message here >>

I'm thinking of a number between 1 and 100...
Your guess? 50
It's lower.
Your guess? 25
It's higher.
Your guess? 35
It's lower.
Your guess? 30
It's higher.
Your guess? 32
It's lower.
Your guess? 31
You got it right in 6 guesses!
Do you want to play again? Y

I'm thinking of a number between 1 and 100...
Your guess? 50
It's higher.
Your guess? 75
It's lower.
Your guess? 65
It's lower.
Your guess? 64
You got it right in 4 guesses!
Do you want to play again? YES
Weeks 7-8

- **Week 7**
  - Scanner
  - Lecture
  - Homework

- **Week 8**

<< your intro

Input file name

Betty Boop:
percent B: [90, 15, 10, 10]
type: ISTJ

Bugs Bunny:
percent B: [20, 45, 15, 55]
type: ESTP
Weeks 9-10

• Week 9: Classes and objects (Ch. 8)
  – fields, methods, constructors, encapsulation
  – lecture programs: Point class; Date class
  – homework: Critters (2D OOP virtual world)

• Week 10: Inheritance/interfaces (Ch. 9); Final exam
  – not tested via homework (exam only)
jGRASP

• decreased emphasis on bulky IDEs (e.g. Eclipse, NetBeans)

• jGRASP
  – simple, basic
  – runs on Win/Mac/Linux
  – excellent debugger
  – data structure visualizations
  – interactions pane
  – soon: JUnit
  – no auto-completion
  – no error underlining
TAs and sections

• weekly meetings w/ instructor
  – what to teach in section
  – common student issues, questions

• "head TA" role

• TA coordinators
  – weekly training

• "community meetings"

CSE 142 Section Handout #4
Cheat Sheet

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math.max(value1, value2)</td>
</tr>
<tr>
<td>Math.min(value1, value2)</td>
</tr>
<tr>
<td>Math.pow(base, exponent)</td>
</tr>
<tr>
<td>Math.round(value)</td>
</tr>
<tr>
<td>Math.sin(value)</td>
</tr>
<tr>
<td>Math.sqrt(value)</td>
</tr>
</tbody>
</table>

1. ("If/Else Mystery 1"). For each call to the following method, indicate what output (Call) you would expect:

public static void mystery1(int n) {
    System.out.print(n + " ");
    if (n > 0) {
        n = n - 5;
    } else { 
        n = n + 7;
    } else {
        n = n * 2;
    }
    System.out.println(n);
}

call
mystery1(8);
mystery1(-3);
mystery1(1);
mystery1(0);

5. Exercise 4.16, p299 ("quadrant"). (Tip: Try testing your solution in our Practice-It web system.)
Write a method called quadrant that accepts as parameters a pair of real numbers representing an \((x, y)\) point and returns the quadrant number for that point. Quadrants are numbered as integers from 1 to 4 with the upper-right quadrant numbered 1 and the subsequent quadrants numbered in a counterclockwise fashion:
Getting help

• support resources for students
  – **lecture programs** very similar to assignment
    • Example: Body Mass Index (BMI) program
  – screen-cast **videos** and **slides** available for each lecture
  – tight integration with **textbook chapter** and case study
  – TA **section** materials directly relate to assignment
  – Introductory Programming Lab (**IPL**) staffed by TAs
Lab hours ("IPL")

- computer lab open 6-7 days per week, ~ noon - 9pm
  - contrast between office hours and IPL hours
- **staffed** by undergrad TAs to answer homework questions
  - TAs are given training to help understand how much help to give!
- TA-written queue/help management [web site](#)
- lists of common questions ([FAQs](#))
Closed lab sessions

- lab handouts for weekly 80-minute labs publicly available
- read/understand, debug/improve, and write code
- tight integration with textbook and course
Style guidelines

• "internal correctness" (style) graded strictly, weighted heavily
  – 50% of homework grade
  – Why?

• resources to students
  – online style guidelines, examples
  – detailed feedback from grader
  – warnings ("minus-0")

Commenting Simple Methods
Let's look at examples of commenting a simple method.

```java
public static void intro() {
    System.out.println("This program...");
    System.out.println("determine...");
    System.out.println("applicant...");
    System.out.println("either SAT...");
}
```

The following comment is too brief. It doesn't add value.
Grade-It

- web-based software for grading homework assignments
  - turnin; scoresheets; annotations; scripts; publishing;
  - data/stats analysis; consistency checks; TA management
Academic integrity

- individual homework assignments (not pairs)
- web software to check for similarity using MOSS
Practice-It

- web-based software for practicing small Java coding problems
  - how it's used in our course; current features/development
Other tools and resources

• Output comparison \textbf{(diff) tool}
• \textbf{Indentation tool}

• Grading gripe submission form

• \textbf{optional sessions}
  – \textbf{Python programming} (also \textbf{CS2})
  – CS "\textbf{exploration sessions}"

• "\textbf{CS is Cool Stuff}" blog

• future: \textbf{Twitter}, Share-It
Conclusion

• (almost) everything we use is available online
  – http://www.cs.washington.edu/142/
  – http://www.buildingjavaprograms.com/

  – assignment/lab/section solutions available to instructors
  – other past assignment specs and ideas (with solutions) available
  – testing/grading scripts, grading criteria
  – past exams, test bank
  – comprehensive lecture slides and notes for CS1 and CS2

• goal: integrated course "ecosystem", course-in-a-box